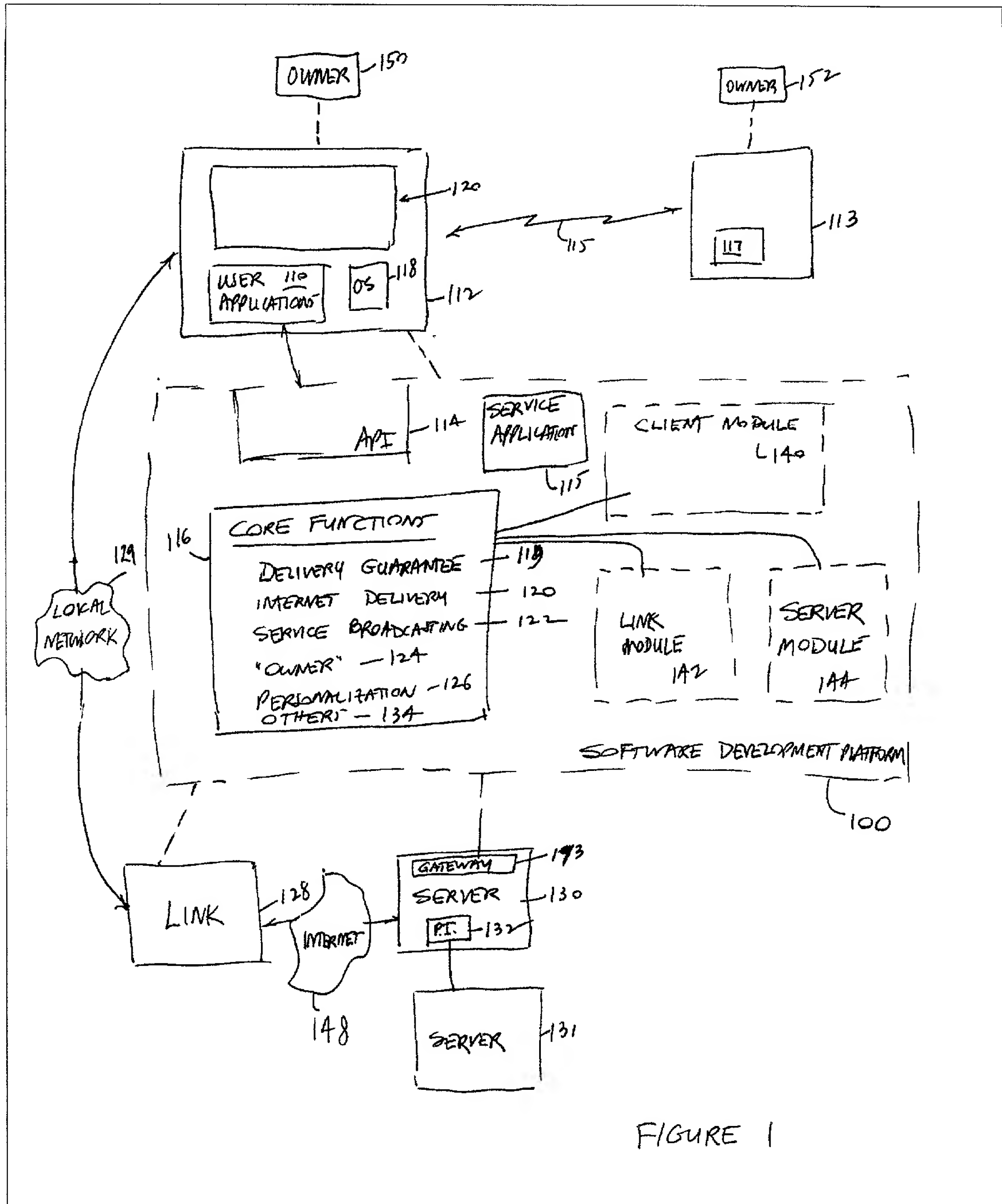


FIGURES



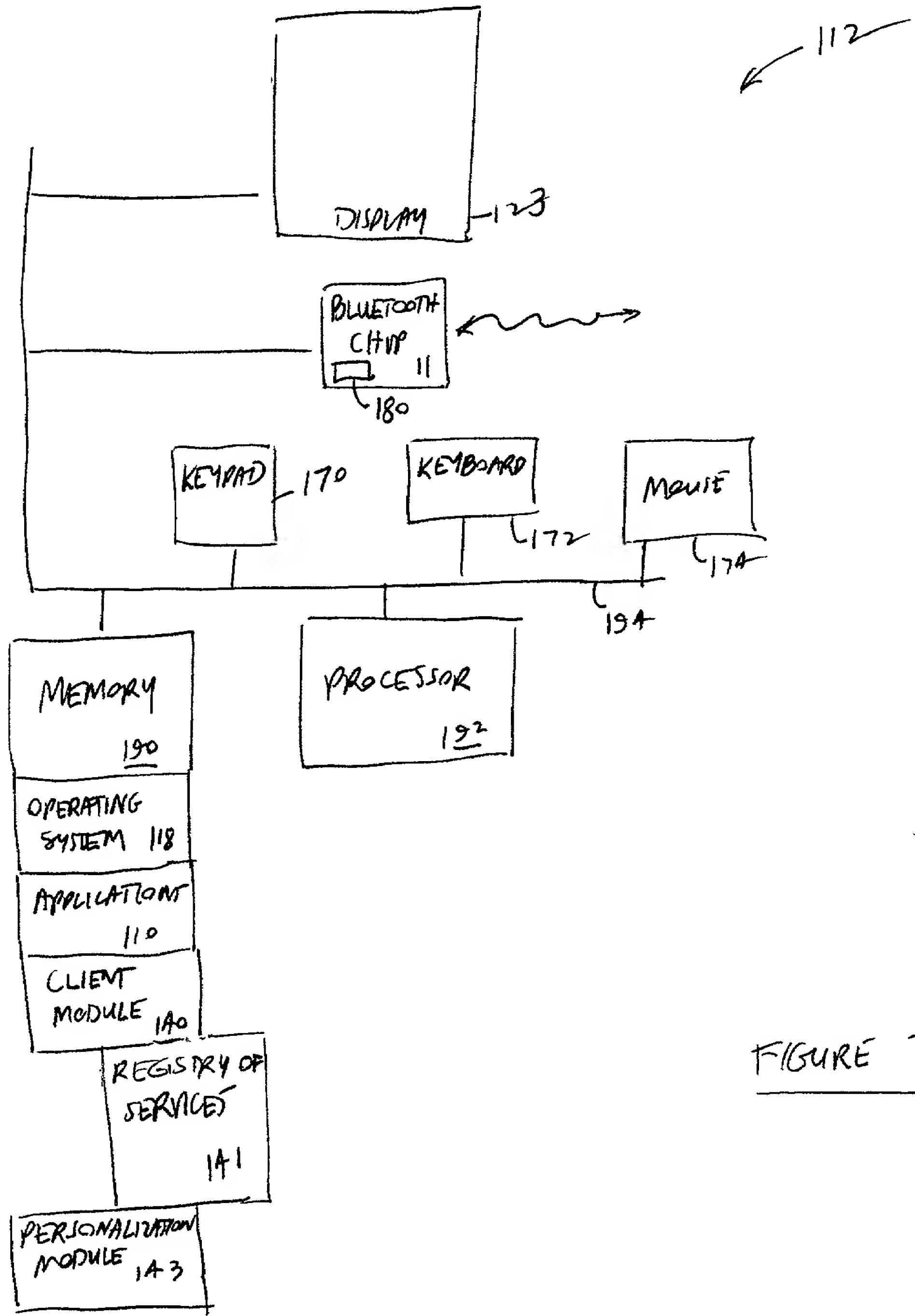


FIGURE 2

FIG. 3A

- LocalService {
 char * serviceName;
 int receiveWhenRunning;
 int receiveWhenNotRunning;
 int running;
 int maxNumOfMessagesToStore;
 char * exp_fld1;
 char * exp_fld2;
 char * exp_fld3;
 char * exp_fld4;
 char * exp_fld5;
};

FIG. 3B

- RemoteService {
 char * serviceName;
 char * userName;
};

FIG. 3C

- KMessage {
 char * serviceName;
 char * date;
 char * recipient;
 char * sender;
 char * messageBody;
};

FIG.
4A

Name	sendMessage
Arguments	char * to, char * serviceName, char * data
Return Values	int err
Description	This function provides sending capabilities so that messages or any kind of unformatted text can be sent between Bluetooth devices. Reception of the text is guaranteed, because even when the devices are not within range, the text is stored and communicated via an Internet connection. If a user is logged in to more than one device simultaneously, the message/text will be sent to both devices at the same time.

FIG.
4B

Name	getMessages
Arguments	struct KMessage * message
Return Values	int err
Description	The getMessages function retrieves all messages or any other unformatted text sent from another Bluetooth device. It returns the data in the KMessage data structure. If a user is logged in to more than one device simultaneously, the message/text will be received from both devices at the same time.

FIG.
4C

Name	getMessage
Arguments	struct KMessage * message
Return Values	int err
Description	The getMessage function retrieves just one message or any other unformatted text sent from another Bluetooth device. It returns the data in the KMessage data structure. If a user is logged in to more than one device simultaneously, the message/text will be received from both devices at the same time.

FIG.
4D

Name	getSurroundingServices
Arguments	struct RemoteService areaServices []
Return Values	int err
Description	This function returns an array of mappings of users and services available on that user's device. This information was previously stored in a database termed the registry, which is a list of devices within range of a Bluetooth device.

FIG.
4E

Name	AddService
Arguments	char * serviceName
Return Values	int err
Description	This function adds a service entry to the registry.

FIG.
4F

Name	RemoveService
Arguments	char * serviceName
Return Values	int err
Description	This function removes a service entry from the registry.

FIG.
4G

Name	changePMTdata
Arguments	<<waiting to hear about PMT API>>
Return Values	int err
Description	A function that allows users to update their personal PMT data and preferences using their particular devices. If the device is not within Bluetooth range of an Internet connection, it will store these update preferences, and make changes within the permanent PMT upon coming into contact with an Internet connection.

FIG.
4H

Name	GetPMTdata
Arguments	char * user
Return Values	int err
Description	Allows a service to get the PMT data of a particular user from the PMT database. If the service cannot reach the PMT database, the information comes from the local storage on the device of the user. Only information that is designated as shared or public data will be retrieved.

FIG.
4I

Name	ChangePMTpermissions
Arguments	<<waiting to hear about PMT API>>
Return Values	int err
Description	This function allows a user to change his PMT permissions from his device.

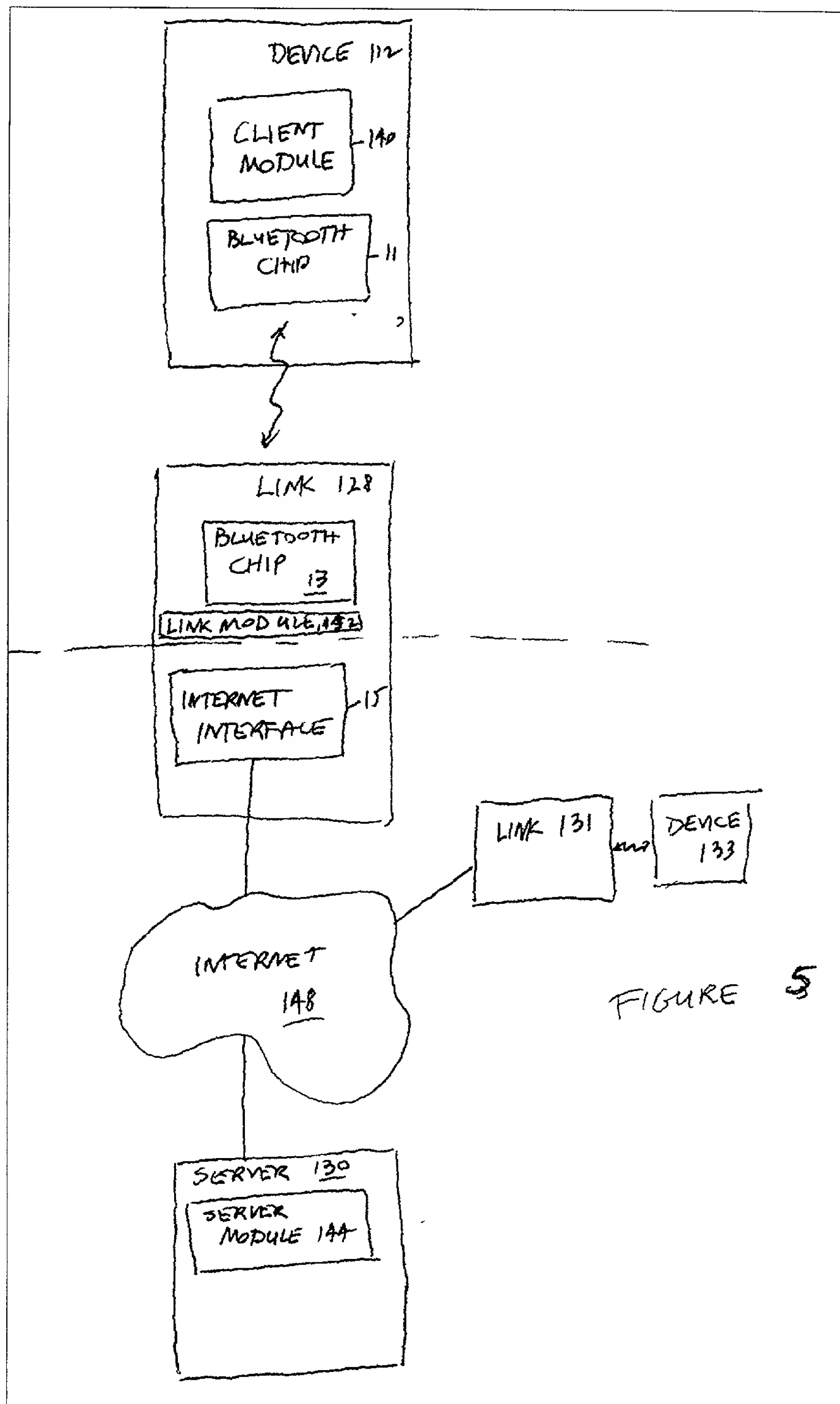


FIGURE 5

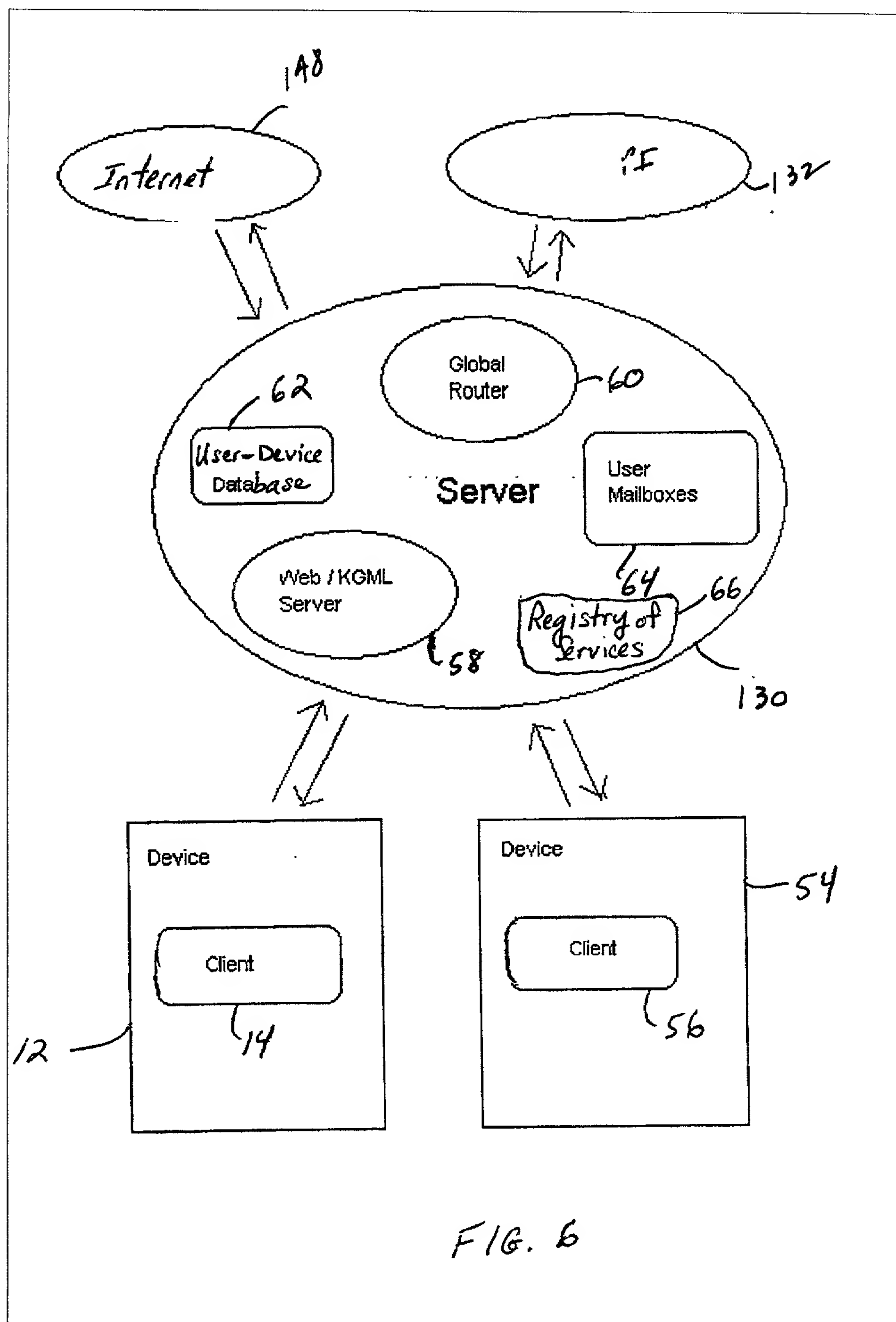


FIG. 6

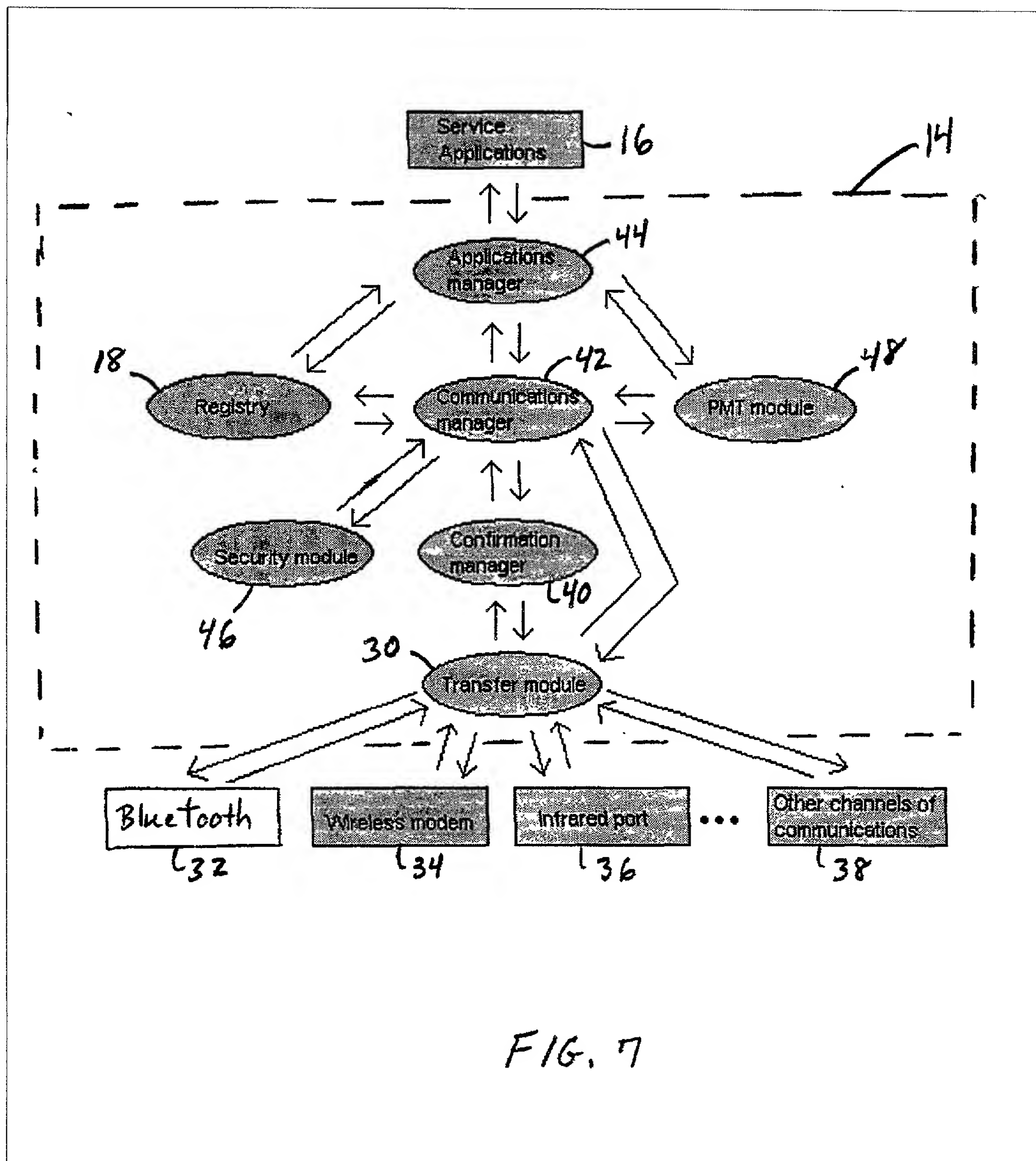


FIG. 7

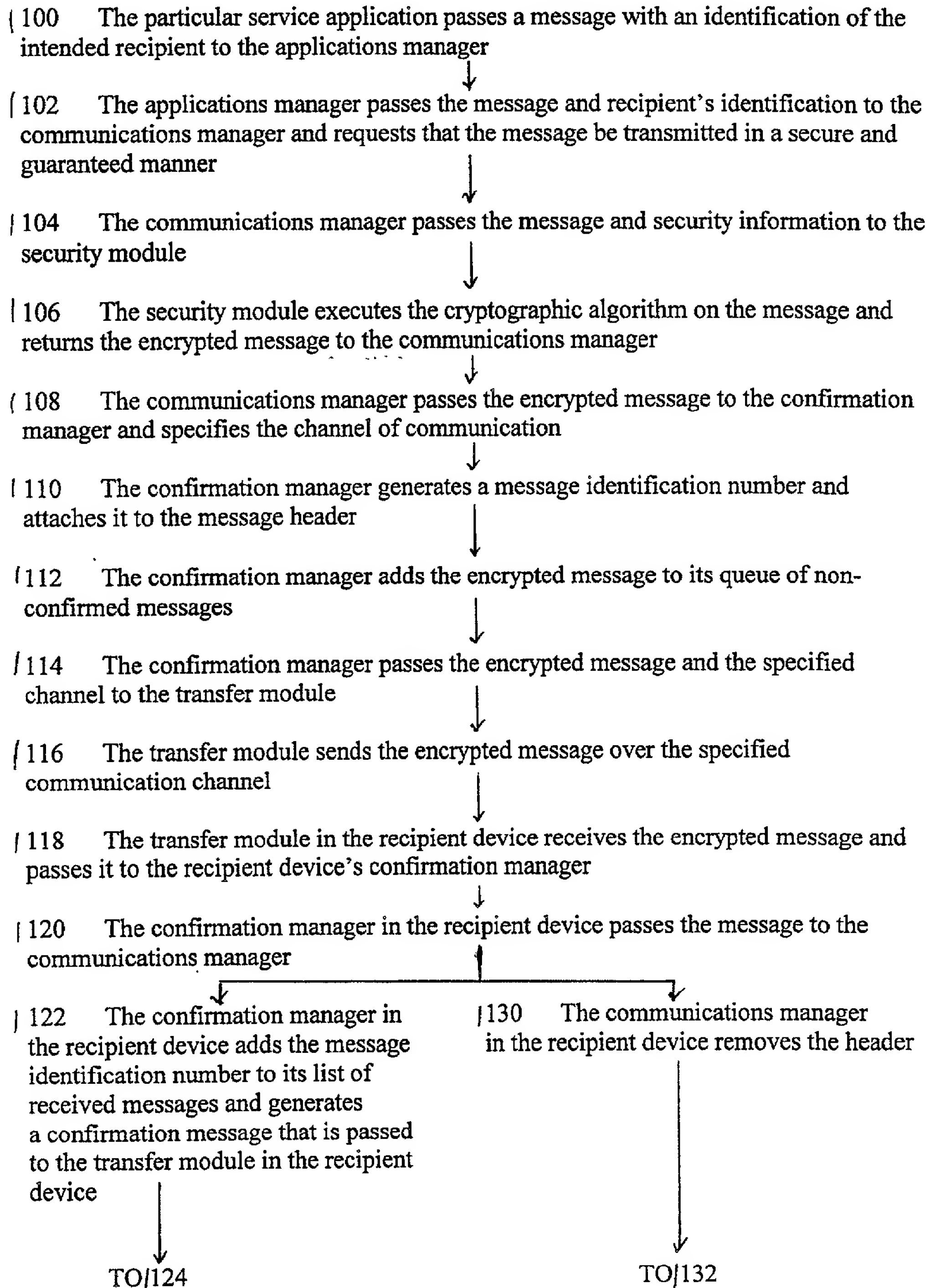


FIG. 8A

| 124 The transfer module sends the confirmation message over the same communication channel over which the original message arrived

| 126 The transfer module in the sending device receives the confirmation message and passes it to the confirmation manager

| 128 The confirmation manager removes the original message from its queue of messages that are awaiting confirmation

| 132 The communication manager passes the encrypted, digitally signed message to the security module in the recipient device

| 134 The security module responds with the sender's identity

| 136 The communications manager in the recipient device adds the decrypted message to its message queue

| 138 The communications manager notifies the user and service application according to the settings in the registry of the recipient device

| 140 The service application requests the message from the communications manager through the applications manager in the recipient device

| 142 The communications manager passes the next message marked for the intended application from the message queue to the service application

| 144 The communications manager deletes the message from its message queue

FIG. 8B

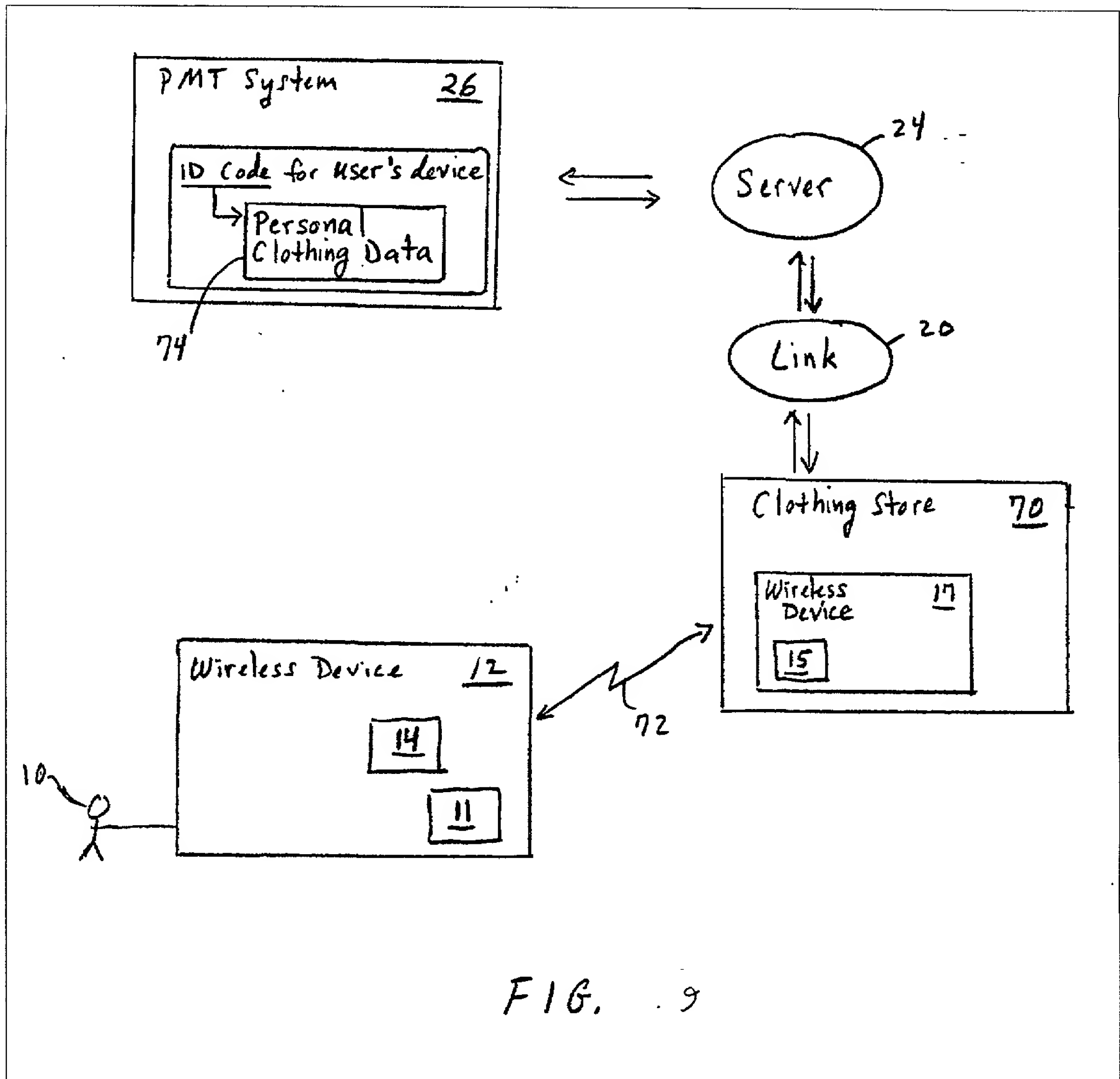
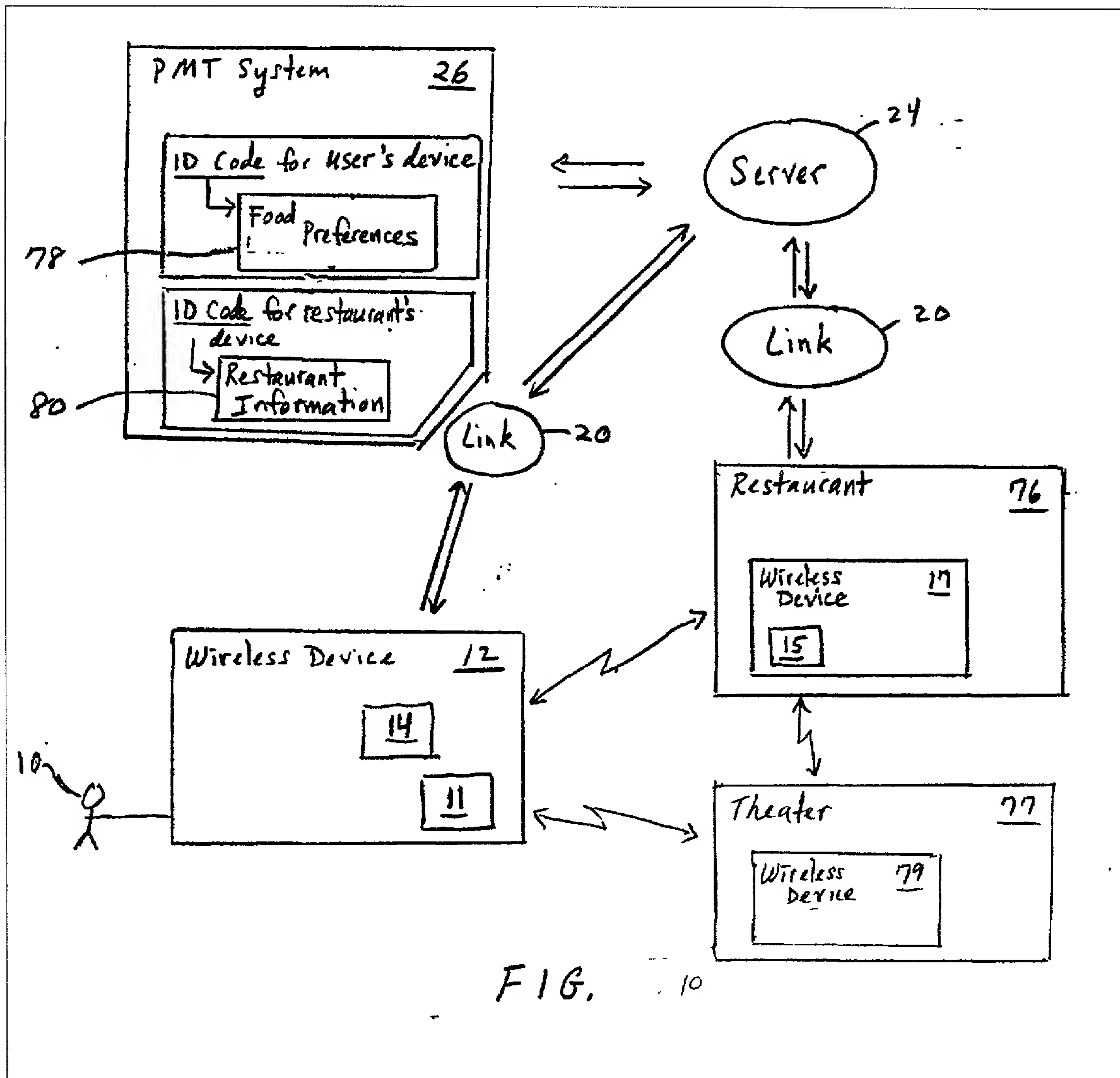
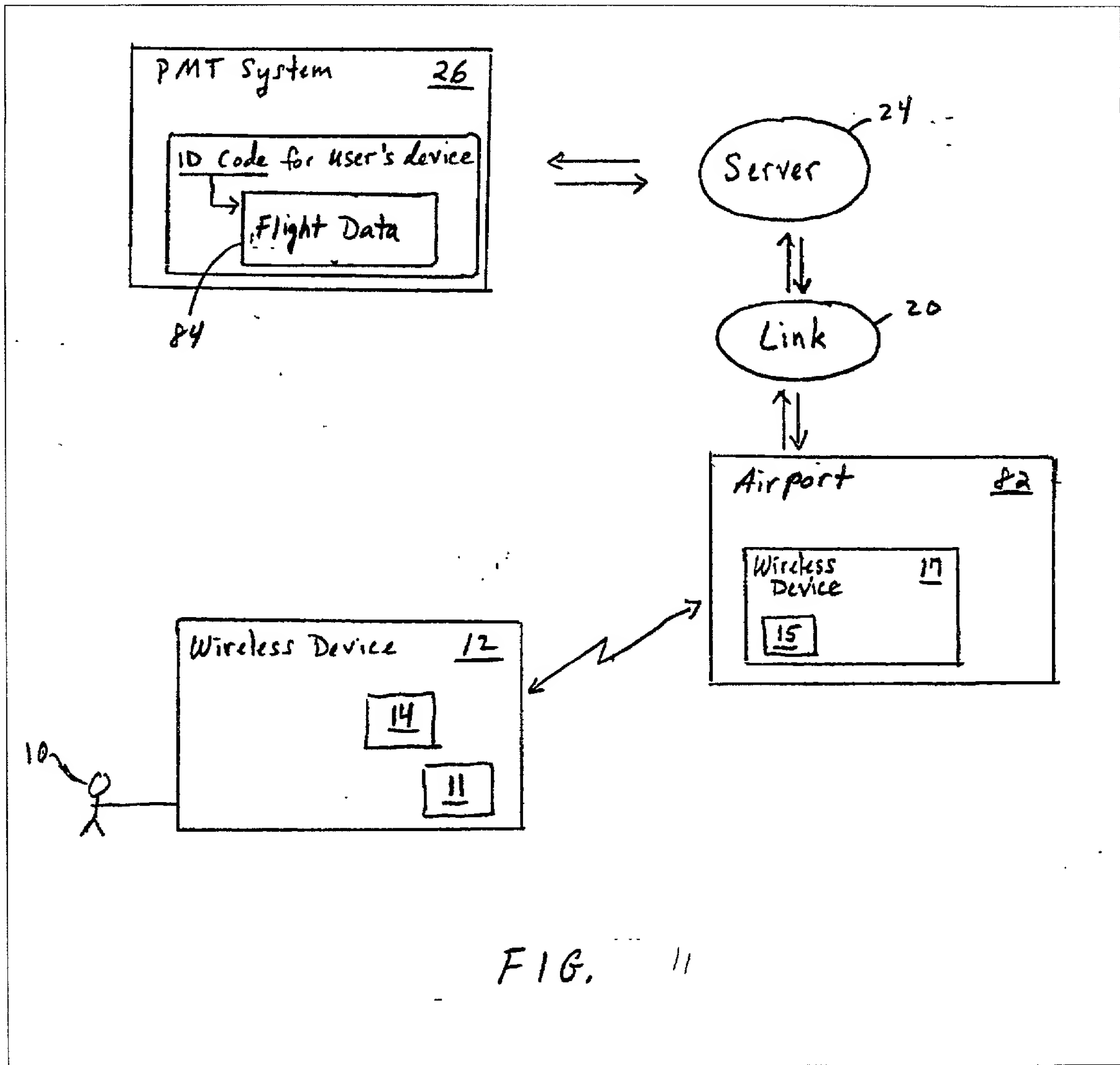


FIG. 9





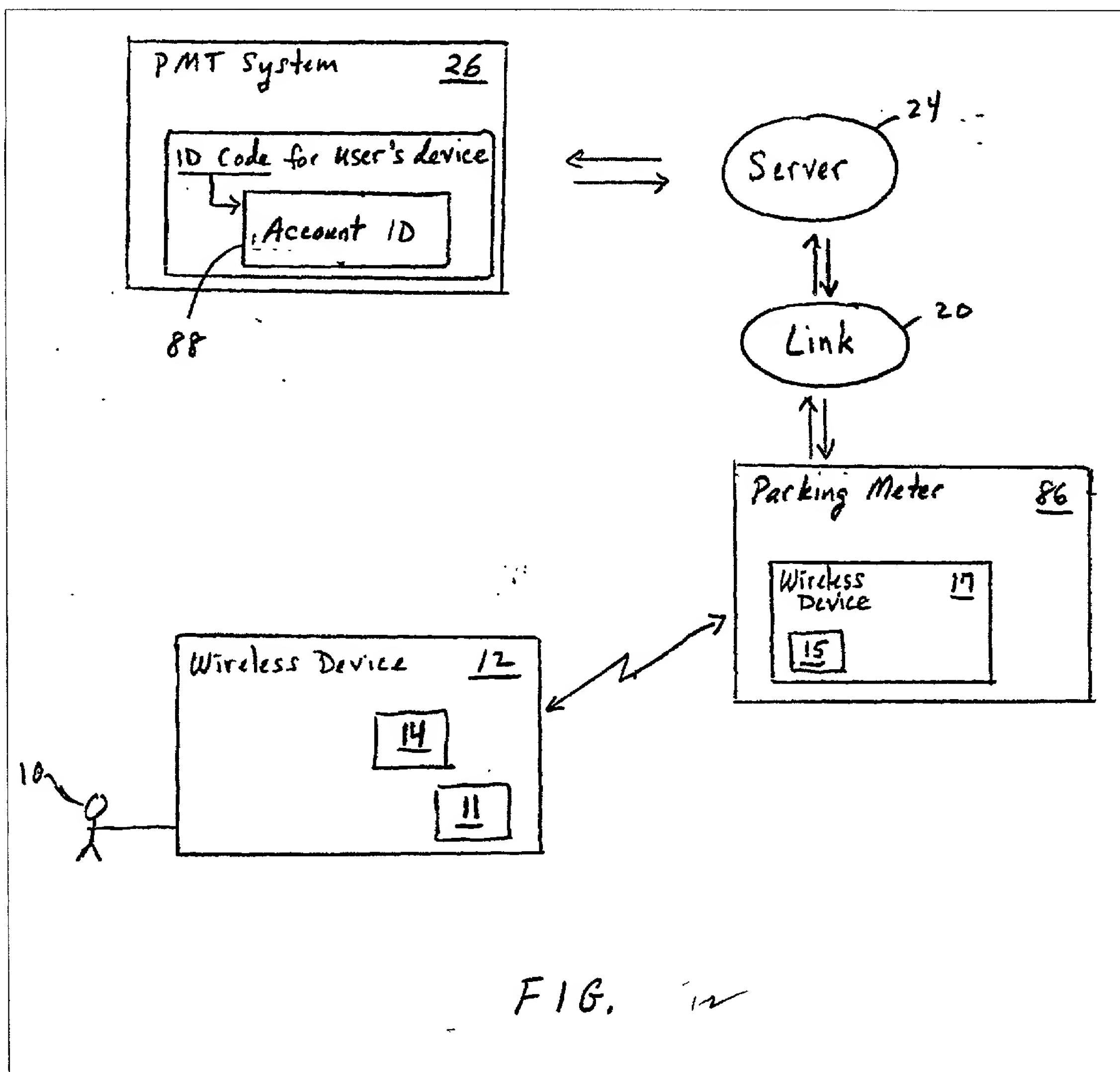
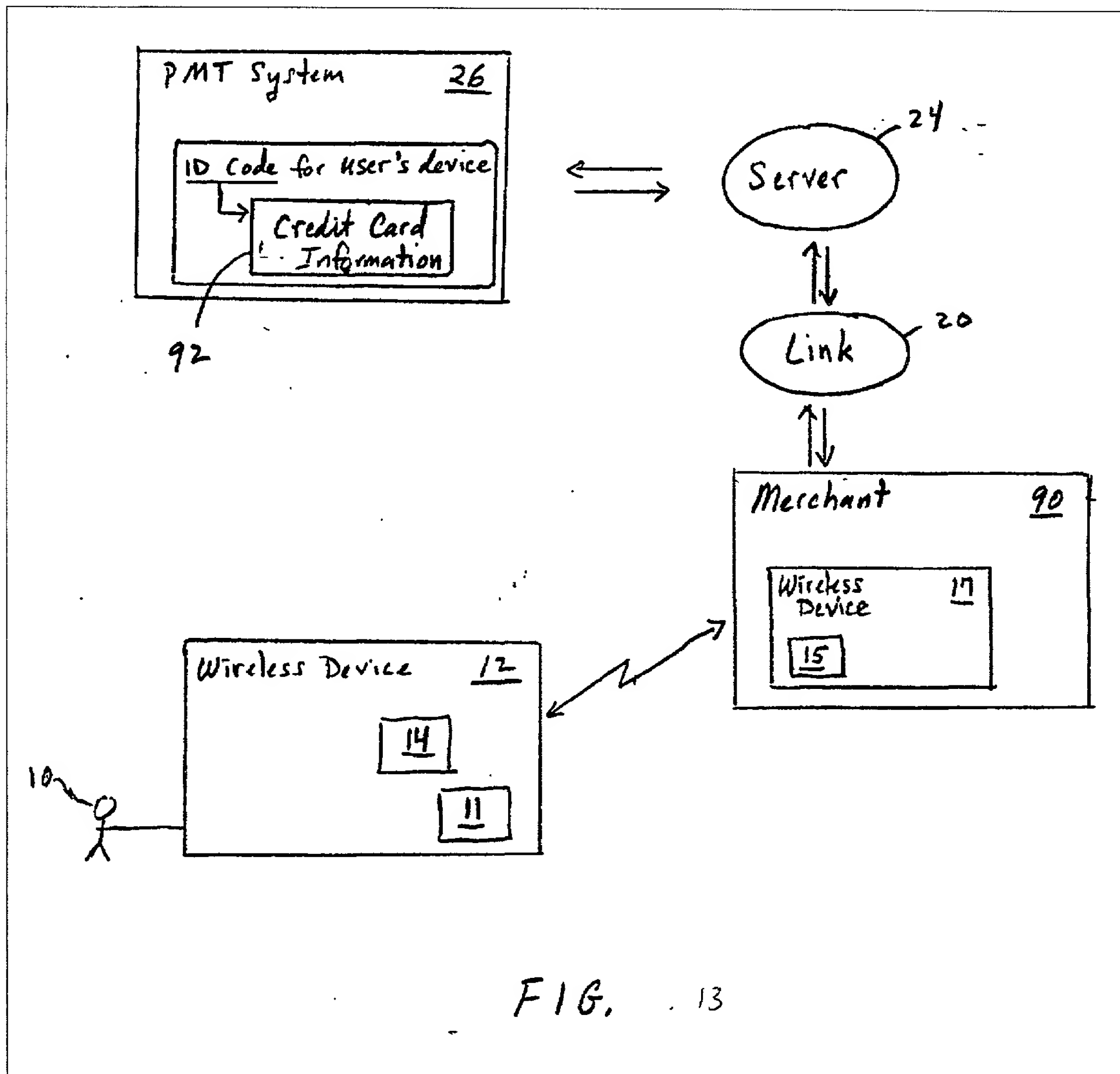


FIG. 12



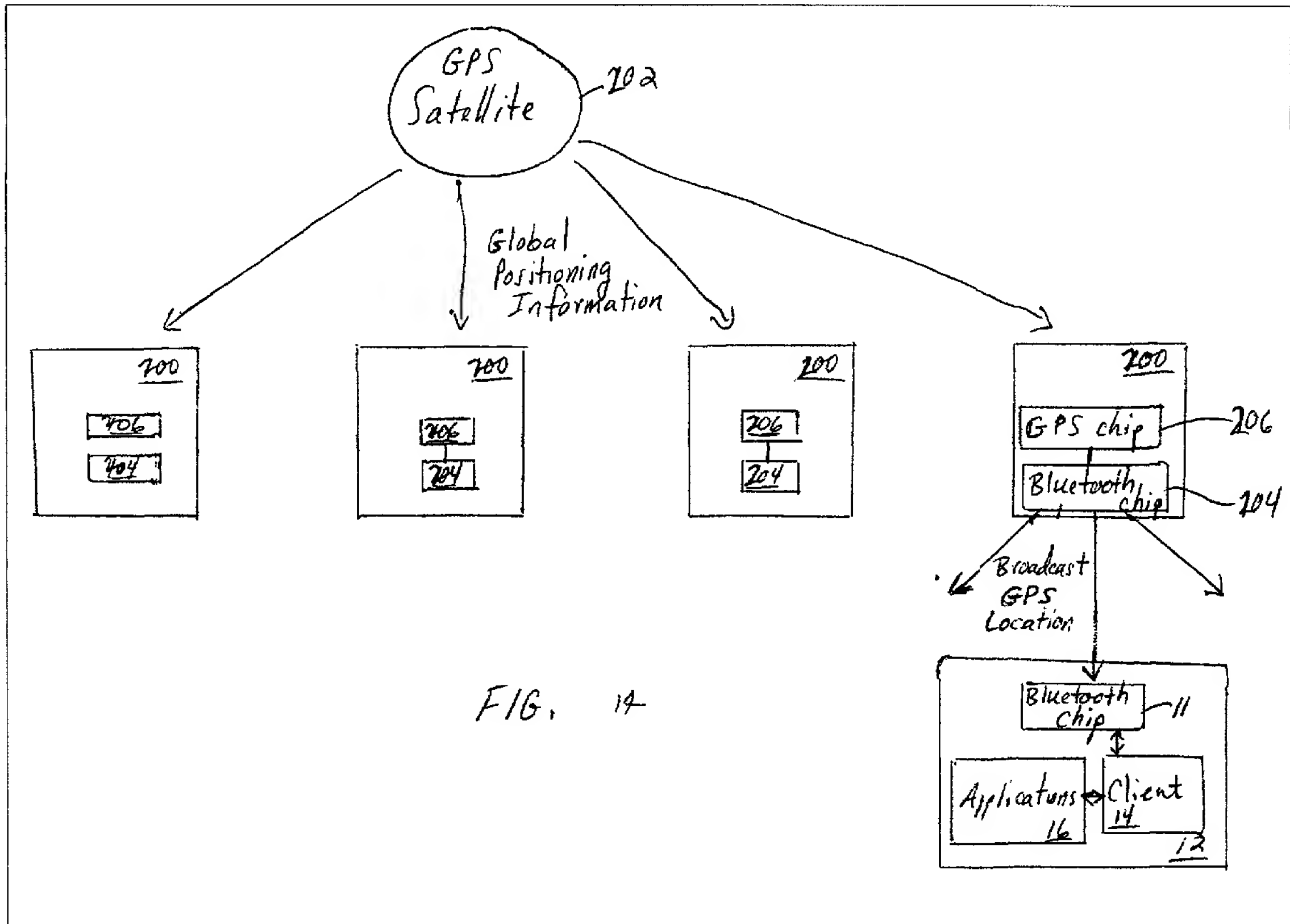


FIG. 14